Chat Server Administration Guide

Configuring a secure connection to Cassandra

5/7/2025

Contents

Configuring a secure connection to Cassandra

The SSL communication mode between Chat Server and Cassandra nodes is optional and can be configured in the **[encryption]** section of the Chat Server Cassandra RAP object.

> **Important**
>
> If a shared Cassandra ring is used, the impact of your settings on other Cassandra-dependent components should be verified prior to making changes.

The following examples assume that:

- The Cassandra cluster consists of two nodes, node1 and node2, running on hosts with IP addresses 172.21.80.85 and 135.225.58.181.
- All passwords in this example are "genesys".
- The java *keytool* and *openssl* are available for certificate creation and manipulation.
- Only one Chat Server Cassandra RAP is configured for all Chat Servers in the solution, so relative paths to the certificates and keys should be the same on all Chat Server hosts. Should these paths be different, you can configure multiple Chat Server Cassandra RAP objects pointing to the same Cassandra cluster.

Example of certificates creation

Cassandra nodes certificate creation

Create a keystore and generate a node1 certificate.

```
keytool -genkeypair -noprompt -keyalg RSA -keysize 2048 -validity 36500 -alias node1 -keystore keystore1.jks -storepass genesys -keypass
genesys -dname "CN=172.21.80.85, O=Genesys, L=Daly City, ST=California, C=US"
```

Keystore **keystore1.jks** should be accessible by node1 and referred to in section **client_encryption_options** of the **cassandra.yaml** file in node1 configuration.

Create a keystore and generate a node2 certificate.

```
keytool -genkeypair -noprompt -keyalg RSA -keysize 2048 -validity 36500 -alias node2 -keystore keystore2.jks -storepass genesys -keypass
genesys -dname "CN=135.225.58.181, O=Genesys, L=Daly City, ST=California, C=US"
```

Keystore **keystore2.jks** should be accessible by node2 and referred to in section **client_encryption_options** of the **cassandra.yaml** file in node2 configuration.

Creating Client Certificates

Generate a client certificate with a private key.

```
openssl req -x509 -days 365 -subj "/C=US/ST=California/L=Daly City/CN=chatclient" -newkey rsa:2048 -keyout chatclientkey.pem -out
chatclient.pem
```

Copy both output files **chatclientkey.pem** and **chatclient.pem** into each Chat Server host and configure the **client-private-key-file** and **client-certificate-file** accordingly.

Exporting of Cassandra Node Certificates

Export node1 certificate:

```
keytool -exportcert -rfc -noprompt -alias node1 -keystore keystore1.jks -storepass genesys -file cassandra1.pem
```

Export node2 certificate:

```
keytool -exportcert -rfc -noprompt -alias node2 -keystore keystore2.jks -storepass genesys -file cassandra2.pem
```

Copy the exported node certificates, **cassandra1.pem** and **cassandra2.pem**, to each Chat Server host into the directory that is passed through each Chat Server Cassandra RAP object **trusted-cert-dir** option.

Importing Client Certificates

Import the client certificate into the truststore of node1:

```
keytool -import -file chatclient.pem -alias chatclient -keystore truststore1.jks -storepass
genesys
```

Import the client certificate of the truststore of node2:

```
keytool -import -file chatclient.pem -alias chatclient -keystore truststore2.jks -storepass
genesys
```

Cassandra and Java with Cryptography Extension

Cassandra nodes with client encryption enabled may fail to start unless Java is updated with the Java Cryptography Extension.

1. Download the Java Cryptography Extension (JCE) from Oracle's website.
2. Replace **US_export_policy.jar** and **local_policy.jar** in your JRE Java folder (found in: **\jre7\lib\security** for Windows or **/jre/lib/security/** for Linux-like platforms).
3. Restart Cassandra.

Client encryption with different Cassandra node certificates and client authentication

In **cassandra.yaml** of node1:

```
client_encryption_options:
  enabled: true
  keystore: <path-to-keystore>/keystore1.jks
  keystore_password: genesys ## The password you used when generating the keystore.
  truststore: <path-to-truststore>/truststore1.jks
  truststore_password: genesys ## The password you used when generating the truststore.
  require_client_auth: true
```

In **cassandra.yaml** of node2:

```
client_encryption_options:
  enabled: true
  keystore: <path-to-keystore>/keystore2.jks
  keystore_password: genesys ## The password you used when generating the keystore.
  truststore: <path-to-truststore>/truststore2.jks
  truststore_password: genesys ## The password you used when generating the truststore.
  require_client_auth: true
```

Cassandra RAP, section encryption:

```
enbabled=true
trusted-cert-dir=<Path to directory containing cassandra1.pem and cassandra2.pem. The
chatclientkey.pem file should not be placed into this directory.>
client-private-key-file=chatclientkey.pem
```

```
password=genesys ## openssl will prompt for this password to be entered during the
certificate creation
client-certificate-file=chatclient.pem
verify-peer-cert=true
verify-peer-identity=true
```

Using cqlsh with SSL encryption

Use the following directions to configure the **cqlshrc** configuration file.The following examples assume that all relevant .pem files are copied into the local C:\certs\ directory.

1. Copy **cqlshrc.sample** from the ~/conf directory to another location, for example **C:\certs\ directory**.
2. Rename the file to **cqlshrc.conf**.
3. Modify the following sections to be consistent with the encryption configuration shown above:

```
[authentication]
;username = fred
;password = !!bang!!$
;; We assumed no user name or password is set in the Cassandra example

[cql]
version = 3.2.0
;; it would not connect with lower version

[connection]
hostname = 172.21.80.85
;; this is node1 of our example
port = 9042
;; we assume the port is default
factory = cqlshlib.ssl.ssl_transport_factory

[ssl]
certfile = C:\certs\cassandra1.pem
;; the certificate of node 1
validate = true
;; assume that we want to validate the node, optional
userkey = C:\certs\chatclientkey.pem
;;if client auth is required on cassandra
usercert = C:\certs\chatclient.pem
;;if client auth is required on cassandra

[certfiles]
172.21.80.85 = C:\certs\cassandra1.pem
;; the cert for node1
135.225.58.181 = C:\certs\cassandra2.pem
;; the cert for node2
```

Start cqlsh with the following command:

```
cqlsh --ssl --cqlshrc=C:\certs\cqlshrc.conf
```

Cqlsh shell should connect to node1 using the configured SSL.

Client encryption with a single Cassandra node certificate and client authentication

In **cassandra.yaml** of node1:

```
client_encryption_options:
```

```
  enabled: true
  keystore: <path-to-keystore>/keystore1.jks
  keystore_password: genesys ## The password you used when generating the keystore.
  truststore: <path-to-truststore>/truststore1.jks
  truststore_password: genesys ## The password you used when generating the truststore.
  require_client_auth: true
```

In **cassandra.yaml** of node2:

```
client_encryption_options:
  enabled: true
  keystore: <path-to-keystore>/keystore1.jks
  keystore_password: genesys ## The password you used when generating the keystore.
  truststore: <path-to-truststore>/truststore2.jks
  truststore_password: genesys ## The password you used when generating the truststore.
  require_client_auth: true
```

Cassandra RAP, section encryption

```
enbabled=true
trusted-cert-dir=<Path to directory containing cassandra1.pem. The chatclientkey.pem file
should not be placed into this directory.>
client-private-key-file=chatclientkey.pem
password=genesys ## openssl will prompt for this password to be entered during the
certificate creation
client-certificate-file=chatclient.pem
verify-peer-cert=true
verify-peer-identity=false
```

Client encryption with a single Cassandra node certificate and no client authentication

In **cassandra.yaml** of node1:

```
client_encryption_options:
  enabled: true
  keystore: <path-to-keystore>/keystore1.jks
  keystore_password: genesys ## The password you used when generating the keystore.
  truststore: <path-to-truststore>/truststore1.jks
  truststore_password: genesys ## The password you used when generating the truststore.
  require_client_auth: false
```

In **cassandra.yaml** of node2:

```
client_encryption_options:
  enabled: true
  keystore: <path-to-keystore>/keystore1.jks
  keystore_password: genesys ## The password you used when generating the keystore.
  truststore: <path-to-truststore>/truststore2.jks
  truststore_password: genesys ## The password you used when generating the truststore.
  require_client_auth: false
```

Cassandra RAP, section encryption

```
enbabled=true
trusted-cert-dir=<Path to directory containing cassandra1.pem>
client-private-key-file=
password= ## empty
client-certificate-file=
verify-peer-cert=true
verify-peer-identity=false
```

ECDHE Cipher Suite Support

When the Java version used does not support ECDHE cipher suite, the cipher_suites option of the **client_encryption_options** section in **cassandra.yaml** file must be modified to exclude cipher suites prefixed with **TLS_ECDHE_**. For example:

```
cipher_suites:
[TLS_RSA_WITH_AES_128_CBC_SHA,TLS_RSA_WITH_AES_256_CBC_SHA,TLS_DHE_RSA_WITH_AES_128_CBC_SHA,TLS_DHE_RSA_WITH_AES_256_CBC_SHA]
 #,TLS_ECDHE_RSA_WITH_AES_128_CBC_SHA,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA]
```